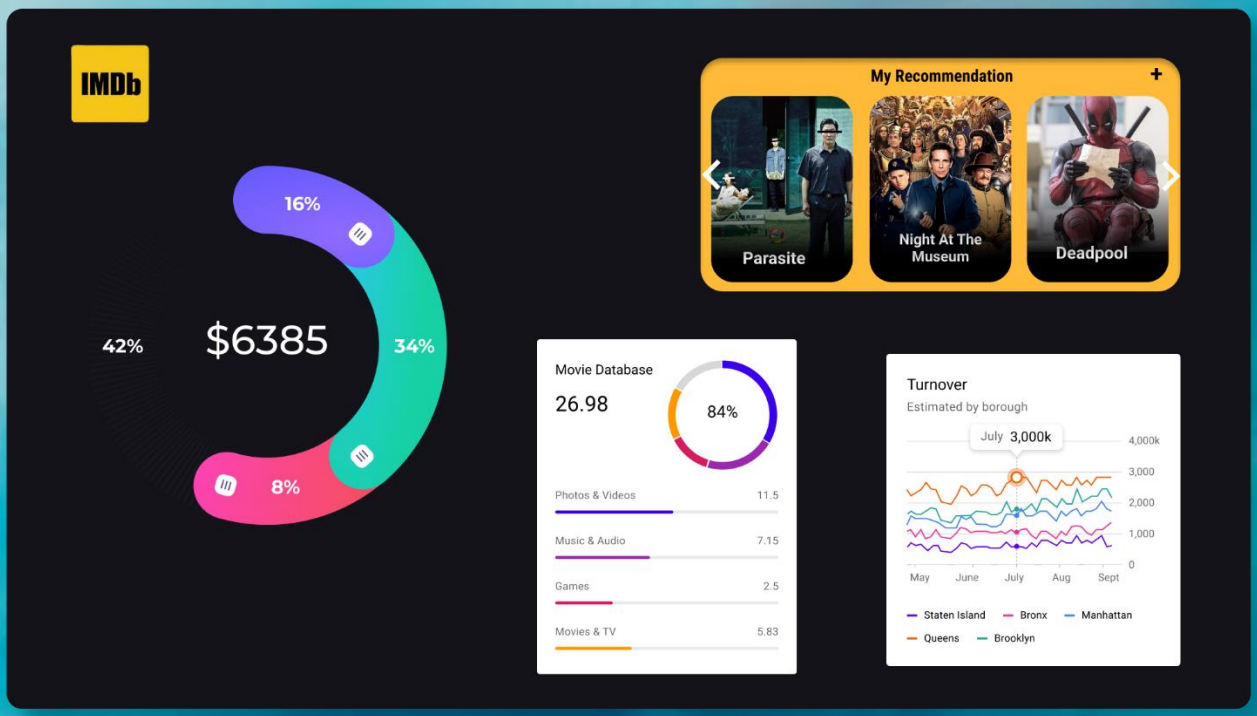


IMDB Movie Analysis

FINAL PROJECT-1
(PROJECT – 5)

Surya Atrish | Data Analytics | 31/08/22

Project Description

For the Final Project, I am provided with dataset having various columns of different IMDB Movies. I am required to Frame the problem. For this task, I need to define a problem I want to shed some light on.

We can do this by asking 'What?' This is where I frame the problem i.e. What is the problem?

We use these questions to guide our thinking:

What do you see happening?
What is your hypothesis for the cause of the problem? (This will be broadly based on intuition initially)
What is the impact of the problem on stakeholders?
What is the impact of the problem not being solved?

Answering these questions will help define a problem we are trying to solve and will allow us to find the right data to solve it.

Once we have defined a problem, we clean the data as necessary, and use our Data Analysis skills to explore the data set and derive insights.

We make sure to use 5 Whys Analysis in your analysis and use this to create a report which conveys a data story.

Once we have framed the problem and gathered initial insights from the data, we can ask the following questions as we dig deeper into our analysis.

What do we see happening?
What are the specific symptoms of the problem?
What is our hypothesis for the cause of the problem?

Five 'Whys' approach

Once we have the problem better defined, we can use 5 Whys technique to determine its root cause by repeatedly asking the question “Why”.

It's also called the Root Cause Analysis, developed by Sakichi Toyoda, founder of Toyota Industries. Here’s an example of how this technique could be used to figure out the cause of the following problem: A business went over budget on a recent project.

Q: “Why did we go over budget on our project?”
A: It took much longer than we expected to complete.

Q: “Why did it take longer than expected to complete?”
A: We had to redesign several elements of the product.

Q: “Why did we have to redesign elements of the product?”
A: Features of the product were confusing to use.

Q: “Why were the features of the product confusing to use?”
A: We made incorrect assumptions about what users wanted.

Q: “Why did we make incorrect assumptions about what users wanted?”
A: Our user experience research team didn’t ask effective questions.

As we see above, what looked like a budgeting problem turned out to be a problem with the user experience team not working effectively.

While asking Why is easy, what we're interested in is the answer. Each time we answer why, the next time gets more difficult as we must think deeper behind the reasons for this. As we ask why, we may find that you have multiple answers for the same question.

Results and Insights

After downloading the dataset, I used MySQL Workbench 8.0 CE(Table movies in Schema imdb) and Microsoft Excel to answer the below questions:

1. **Cleaning the data:** This is one of the most important steps to perform before moving forward with the analysis. Use your knowledge learned till now to do this. (Dropping columns, removing null values, etc.)
   Your task: Clean the data
   Solution:
   a. First, I analyze the dataset in MS Excel to find out that there are 5043 rows, 28 columns and 1 header row containing the column names.
   b. Then, after looking at the given questions, we find out that most columns are not required to find out the solution. So, we remove the following columns:
      i. color
      ii. director_facebook_likes
      iii. actor_1_facebook_likes
      iv. actor_2_facebook_likes
      v. actor_3_facebook_likes
      vi. actor_2_name
      vii. cast_total_facebook_likes
      viii. actor_3_name
      ix. duration

x. facenumber_in_poster
xi. content_rating
xii. country
xiii. movie_imdb_link
xiv. aspect_ratio
xv. plot_keywords

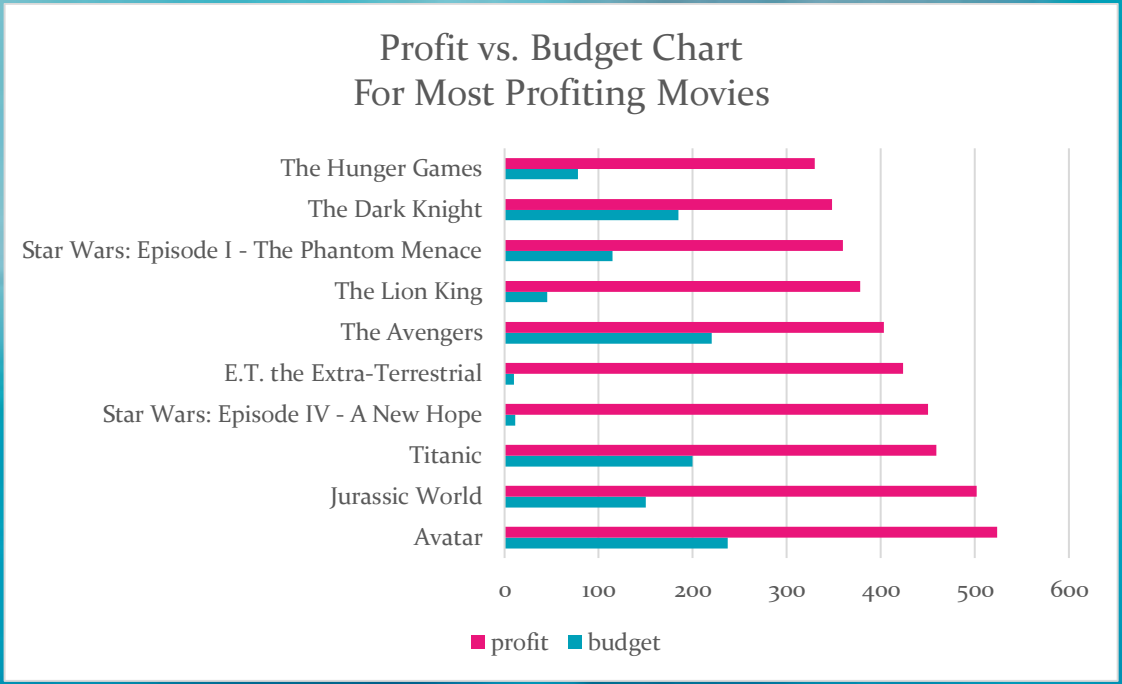
c. After this, only 13 columns are left. Now, we count the number of null/blank values in each column using the formula =COUNTBLANK(A2:A5044) and drag it over all columns. Then, we calculate the null value percentage using the formula =A5046/5043*100 and drag it over all columns to find out that 'gross' and 'budget' column have very high null value percentages, 17.53 and 9.76 respectively to be exact.
d. Now, we open the dataset in MySQL Workbench 8.0 CE and drop all the rows where gross or budget is NULL using the query
DELETE FROM imdb.movies
WHERE gross IS NULL or budget IS NULL;
e. This reduces most of the null value percentages down to zero. The last important column having high null value percentage is 'language', where we safely replace NULL with 'English' to reduce all null values.
f. Next, we remove all the duplicate rows from the table.
g. After all this, the total number of rows retained is 3789.

2. **Movies with highest profit:** Create a new column called profit which contains the difference of the two columns: gross and budget. Sort the column using the profit column as reference. Plot profit (y-axis) vs budget (x- axis) and observe the outliers using the appropriate chart type.
Your task: Find the movies with the highest profit?
Solution:
a. Now, as the values in gross and budget columns are very large, we divide the values by million and round them off to 2 decimal places.
b. Next, we create a 'profit' column as gross – budget for each row using this query:

```
SELECT
   director_name, movie_title, round(budget, 0) as budget,
round(gross - budget, 0) as profit
FROM
   movies
ORDER BY gross - budget DESC
LIMIT 10
```

c. With this query, we found out that the top 10 most profiting movies are:
i. Avatar

ii. Jurassic World
iii. Titanic
iv. Star Wars: Episode IV - A New Hope
v. E.T. the Extra-Terrestrial
vi. The Avengers
vii. The Lion King
viii. Star Wars: Episode I - The Phantom Menace
ix. The Dark Knight
x. The Hunger Games



3. **Top 250:** Create a new column IMDb_Top_250 and store the top 250 movies with the highest IMDb Rating (corresponding to the column: imdb_score). Also make sure that for all of these movies, the num_voted_users is greater than 25,000. Also add a Rank column containing the values 1 to 250 indicating the ranks of the corresponding films.
Extract all the movies in the IMDb_Top_250 column which are not in the English language and store them in a new column named Top_Foreign_Lang_Film. You can use your own imagination also!
Your task: Find IMDB Top 250 and Top Non English Movies

Solution:

a. For the first part we look for all the movies having the highest imdb scores with no. of user votes higher than 25000 using the below query:

```
SELECT
   row_number() over(order by imdb_score DESC,
num_voted_users DESC) as ranking,
   imdb_score, num_voted_users, movie_title AS
IMDb_Top_250, language
FROM
   imdb.movies
WHERE
   num_voted_users > 25000
LIMIT 250;
```

b. Next, out of these movies, we search for all the Top 250 Non- English movies using the below the query:

```
SELECT
 row_number() over(order by imdb_score DESC,
num_voted_users DESC) as ranking,
  imdb_score, num_voted_users, movie_title AS
IMDb_Top_250, language
FROM
  imdb.movies
WHERE
   num_voted_users > 25000
LIMIT 250;
```

4. **Best Directors:** Group the column using the director_name column. Find out the top 10 directors for whom the mean of imdb_score is the highest and store them in a new column top10director. In case of a tie in IMDb score between two directors, sort them alphabetically.
   Your task: Find the best directors.
   Solution:
   a. For this question, we sort the names of directors in descending order of their average imdb scores using the below query:

```
SELECT
   director_name AS top10director,
   round(AVG(imdb_score), 2) AS avg_score
FROM
   imdb.movies
GROUP BY director_name
ORDER BY avg_score DESC , director_name
LIMIT 10
```

   b. This query showed us the top 10 directors as follows:

top10director	avg_score
Charles Chaplin	8.6
Tony Kaye	8.6
Alfred Hitchcock	8.5
Damien Chazelle	8.5

Majid Majidi	8.5
Ron Fricke	8.5
Christopher Nolan	8.43
Sergio Leone	8.43
Asghar Farhadi	8.4
Marius A. Markevicius	8.4

5. **Popular Genres:** Perform this step using the knowledge gained while performing previous steps.
   Your task: Find popular genres
   Solution:
   a. For this question, we first split the genres of each movie into individual genres into 8 separate levels of genres. Then, we make a list of all genres by uniting all levels of genres and then count the popularity of each genre.
   b. The query is as follows:

```
With d8 as (SELECT *,
      (CASE WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres,
'|', 8), '|', -1) = genre_1 THEN NULL
       WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 8),
'|', -1) = genre_2 THEN NULL
       WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 8),
'|', -1) = genre_3 THEN NULL
       WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 8),
'|', -1) = genre_4 THEN NULL
       WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 8),
'|', -1) = genre_5 THEN NULL
       WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 8),
'|', -1) = genre_6 THEN NULL
       WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 8),
'|', -1) = genre_7 THEN NULL
       ELSE SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 8),
'|', -1) END) AS genre_8
FROM(SELECT *,
      (CASE WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres,
'|', 7), '|', -1) = genre_1 THEN NULL
       WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 7),
'|', -1) = genre_2 THEN NULL
       WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 7),
'|', -1) = genre_3 THEN NULL
       WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 7),
'|', -1) = genre_4 THEN NULL
       WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 7),
'|', -1) = genre_5 THEN NULL
```

```
        WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 7),
'|', -1) = genre_6 THEN NULL
        ELSE SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 7),
'|', -1) END) AS genre_7
FROM(SELECT *,
      (CASE WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres,
'|', 6), '|', -1) = genre_1 THEN NULL
        WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 6),
'|', -1) = genre_2 THEN NULL
        WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 6),
'|', -1) = genre_3 THEN NULL
        WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 6),
'|', -1) = genre_4 THEN NULL
        WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 6),
'|', -1) = genre_5 THEN NULL
        ELSE SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 6),
'|', -1) END) AS genre_6
FROM(SELECT *,
      (CASE WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres,
'|', 5), '|', -1) = genre_1 THEN NULL
        WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 5),
'|', -1) = genre_2 THEN NULL
        WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 5),
'|', -1) = genre_3 THEN NULL
        WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 5),
'|', -1) = genre_4 THEN NULL
        ELSE SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 5),
'|', -1) END) AS genre_5
FROM(SELECT *,
      (CASE WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres,
'|', 4), '|', -1) = genre_1 THEN NULL
        WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 4),
'|', -1) = genre_2 THEN NULL
        WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 4),
'|', -1) = genre_3 THEN NULL
        ELSE SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 4),
'|', -1) END) AS genre_4
FROM (SELECT *,
      (CASE WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres,
'|', 3), '|', -1) = genre_1 THEN NULL
        WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 3),
'|', -1) = genre_2 THEN NULL
        ELSE SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 3),
'|', -1) END) AS genre_3
```

```
FROM(SELECT *,
      (CASE WHEN SUBSTRING_INDEX(SUBSTRING_INDEX(genres,
'|', 2), '|', -1) = genre_1 THEN NULL
       ELSE SUBSTRING_INDEX(SUBSTRING_INDEX(genres, '|', 2),
'|', -1) END) AS genre_2
FROM (SELECT SUBSTRING(genres, 2) as genres,
       SUBSTRING_INDEX(SUBSTRING_INDEX(SUBSTRING(genres,
2), '|', 1), '|', - 1) AS genre_1
       FROM imdb.movies) d1) d2) d3) d4) d5) d6) d7),
list as (SELECT DISTINCT genre_1 AS genre_list
    FROM ((SELECT genre_1 FROM d8) UNION ALL (SELECT
genre_2 FROM d8)
    UNION ALL (SELECT genre_3 FROM d8) UNION ALL (SELECT
genre_4 FROM d8)
    UNION ALL (SELECT genre_5 FROM d8) UNION ALL (SELECT
genre_6 FROM d8)
    UNION ALL (SELECT genre_7 FROM d8) UNION ALL (SELECT
genre_8 FROM d8)) g12
WHERE genre_1 IS NOT NULL
ORDER BY genre_1)

SELECT genre_list,
       round(AVG(DISTINCT CASE WHEN INSTR(genres, genre_list)
THEN gross ELSE NULL END), 2) AS avg_grossing,
      COUNT(DISTINCT CASE WHEN INSTR(genres, genre_list)
THEN genres ELSE NULL END) AS no_of_movies
FROM movies JOIN list
GROUP BY genre_list
ORDER BY avg_grossing DESC
```

c. This query showed us the popularity of all genres as follows:

genre_list	avg_grossing	no_of_movies
Animation	107.66	92
Adventure	97.78	257
Family	89.59	185
Fantasy	84.92	184
Sci-Fi	82.38	149
Action	75.72	252
Musical	55.49	70
Comedy	49.79	302
Thriller	48.15	202
Mystery	46.3	107
Music	44.9	123
Western	43.72	39

Sport	42.96	45
Romance	42.85	193
War	40.11	73
Crime	38.67	146
History	36.63	61
Drama	36.27	360
Biography	35.84	83
Horror	34.29	92
Documentary	13.38	28
Short	3.93	2
Film-Noir	0.01	1

6. **Most Popular Actor:** Create three new columns namely, Meryl_Streep, Leo_Caprio, and Brad_Pitt which contain the movies in which the actors: 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' are the lead actors. Use only the actor_1_name column for extraction. Also, make sure that you use the names 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' for the said extraction.
   Append the rows of all these columns and store them in a new column named Combined.
   Group the combined column using the actor_1_name column.
   Your task: Find the mean of the num_critic_for_reviews and num_users_for_review and identify the actors which have the highest mean.
   Solution:
   a. The query is as follows:

```
SELECT actor_1_name,
   COUNT(movie_title) AS no_of_movies,
   ROUND(AVG(num_user_for_reviews), 2) AS user_reviews,
   ROUND(AVG(num_critic_for_reviews), 2) AS critic_reviews
FROM ((SELECT actor_1_name, movie_title,
        num_user_for_reviews, num_critic_for_reviews
   FROM imdb.movies
   WHERE actor_1_name = ' Meryl Streep')
   UNION ALL (SELECT actor_1_name, movie_title,
        num_user_for_reviews, num_critic_for_reviews
   FROM imdb.movies
   WHERE actor_1_name = ' Leonardo DiCaprio')
   UNION ALL (SELECT actor_1_name, movie_title,
        num_user_for_reviews, num_critic_for_reviews
   FROM imdb.movies
   WHERE actor_1_name = ' Brad Pitt')) c
GROUP BY actor_1_name
ORDER BY user_reviews DESC , critic_reviews DESC
```

b. This query helped me to find out that Leonardo DiCaprio is the most favorite actor for both the audience and the critics.

actor_1_name	no_of_movies	user_reviews	critic_reviews
Leonardo DiCaprio	20	922.55	322.2
Brad Pitt	17	742.35	245
Meryl Streep	11	297.18	181.45

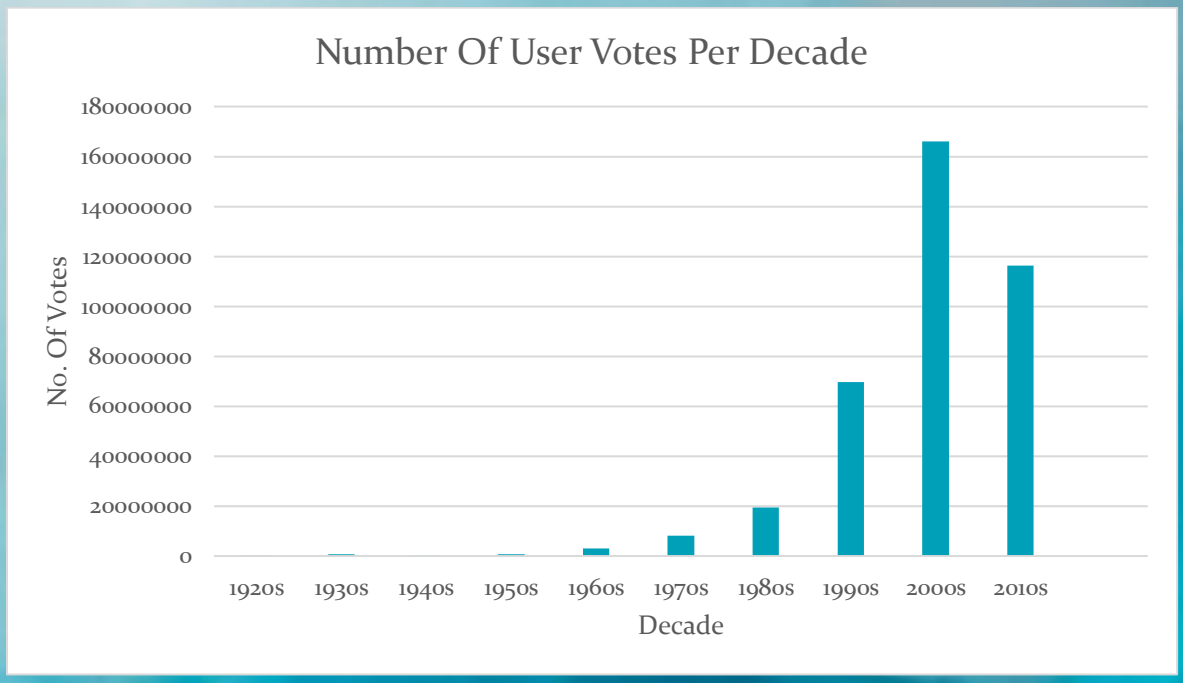
7. **Bar Chart:** Observe the change in number of voted users over decades using a bar chart. Create a column called decade which represents the decade to which every movie belongs to. For example, the title_year year 1923, 1925 should be stored as 1920s. Sort the column based on the column decade, group it by decade and find the sum of users voted in each decade.
Your task: Find the number of user votes per decade.
Solution:
   a. The query is as follows:

```
SELECT
   CONCAT(CONVERT( FLOOR(title_year / 10) * 10 , CHAR),
        's') AS decade,
   SUM(num_voted_users) AS total_votes
FROM
   movies
GROUP BY decade
ORDER BY decade
```

   b. This query helped me to find out the number of user votes per decade, which are as follows:

decade	total_votes
1920s	116392
1930s	804839
1940s	230838
1950s	678336
1960s	2983442
1970s	8269829
1980s	19344380
1990s	69635865
2000s	165976676
2010s	116240375

c. The bar graph is as follows:



Approach

I first created the IMDb movies database for the project using the csv file provided, then assessed the database to remove outliers, null rows, useless columns and generated meaningful insights to help us answer the given questions and complete the project.